中华人民共和国国家知识产权局
STATE INTELLECTUAL PROPERTY OFFICE
OF THE PEOPLE'S REPUBLIC OF CHINA

证 明

CERTIFICATE

本证明之附件是向中国专利局作为受理局提交的下列国际申请副本
THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY OF THE BELOW IDENTIFIED INTERNATIONAL APPLICATION THAT WAS FILED WITH THE CHINESE PATENT OFFICE AS RECEIVING OFFICE

国 际 申 请 号：PCT/CN2005/002386
INTERNATIONAL APPLICATION NUMBER

国 际 申 请 日：30. 12 月 2005(30.12.2005)
INTERNATILNAL FILING DATE

发 明 名 称：VIRTUAL MACHINE TO DETECT MALICIOUS CODE
TITLE OF INVENTION

CERTIFIED COPY OF
PRIORITY DOCUMENT

中华人民共和国国家知识产权局局长
COMMISSIONER OF THE STATE INTELLECTUAL PROPERTY
OFFECE OF THE PEOPLE'S REPUBLIC OF CHINA

二零零六年四月十七日
APRIL 17, 2006

HOME COPY

PCT

REQUEST

The undersigned requests that the present international application be processed according to the Patent Cooperation Treaty.

For receiving Office use only

International Application No. PCT/CN2005 / 002386

International Filing Date 30 · 12月 2005 (30 · 12 · 2005)

RO/CN 中华人民共和国国家知识产权局 PCT International Application

Name of receiving Office and "PCT International Application"

Applicant's or agent's file reference *(if desired) (12 characters maximum)* I05BJ1141

Box No. I TITLE OF INVENTION

VIRTUAL MACHINE TO DETECT MALICIOUS CODE

Box No. II APPLICANT ☐ This person is also inventor

Name and address: *(Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country. The country of the address indicated in this Box is the applicant's State (that is, country) of residence if no State of residence is indicated below.)*

INTEL CORPORATION
2200 Mission College Boulevard
Santa Clara, California 95052
United States of America

Telephone No.

Facsimile No.

Teleprinter No.

Applicant's registration No. with the Office

State *(that is, country)* of nationality: US

State *(that is, country)* of residence: US

This person is applicant for the purposes of: ☐ all designated States ☒ all designated States except the United States of America ☐ the United States of America only ☐ the States indicated in the Supplemental Box

Box No. III FURTHER APPLICANT(S) AND/OR (FURTHER) INVENTOR(S)

Name and address: *(Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country. The country of the address indicated in this Box is the applicant's State (that is, country) of residence if no State of residence is indicated below.)*

PENG, Zhang
Room 402, No.122, Hangxin Road, Shanghai 201105, P. R. China

This person is:

☐ applicant only

☒ applicant and inventor

☐ inventor only *(If this check-box is marked, do not fill in below.)*

Applicant's registration No. with the Office

State *(that is, country)* of nationality: CN

State *(that is, country)* of residence: CN

This person is applicant for the purposes of: ☐ all designated States ☐ all designated States except the United States of America ☒ the United States of America only ☐ the States indicated in the Supplemental Box

☐ Further applicants and/or (further) inventors are indicated on a continuation sheet.

Box No. IV AGENT OR COMMON REPRESENTATIVE; OR ADDRESS FOR CORRESPONDENCE

The person identified below is hereby/has been appointed to act on behalf of the applicant(s) before the competent International Authorities as: ☒ agent ☐ common representative

Name and address: *(Family name followed by given name; for a legal entity, full official designation. The address must include postal code and name of country.)*

IntellecPro China Limited
Suites 902-908, Ping'an Mansion,
23 Jinrong Dajie, Xicheng District,
Beijing 100032, P. R. China

Telephone No. 86-10-66215588

Facsimile No. 86-10-66210771

Teleprinter No.

Agent's registration No. with the Office 11015

☐ **Address for correspondence:** Mark this check-box where no agent or common representative is/has been appointed and the space above is used instead to indicate a special address to which correspondence should be sent.

Box No. V DESIGNATIONS

The filing of this request **constitutes under Rule 4.9(a), the designation** of all Contracting States bound by the PCT on the international filing date, for the grant of every kind of protection available and, where applicable, for the grant of both regional and national patents.

However,

☐ DE Germany **is not designated** for any kind of national protection

☐ KR Republic of Korea **is not designated** for any kind of national protection

☐ RU Russian Federation **is not designated** for any kind of national protection

(The check-boxes above may be used to exclude (irrevocably) the designations concerned in order to avoid the ceasing of the effect, under the national law, of an earlier national application from which priority is claimed. See the Notes to Box No. V as to the consequences of such national law provisions in these and certain other States.)

Box No. VI PRIORITY CLAIM

The priority of the following earlier application(s) is hereby claimed:

Filing date of earlier application *(day/month/year)*	Number of earlier application	Where earlier application is:		
		national application: country or Member of WTO	regional application:* regional Office	international application: receiving Office
item (1)				
item (2)				
item (3)				

☐ Further priority claims are indicated in the Supplemental Box.

The receiving Office is requested to prepare and transmit to the International Bureau a certified copy of the earlier application(s) *(only if the earlier application was filed with the Office which for the purposes of this international application is the receiving Office)* identified above as:

☐ all items ☐ item (1) ☐ item (2) ☐ item (3) ☐ other, see Supplemental Box

* *Where the earlier application is an ARIPO application, indicate at least one country party to the Paris Convention for the Protection of Industrial Property or one Member of the World Trade Organization for which that earlier application was filed (Rule 4.10(b)(ii)):*

.....

Box No. VII INTERNATIONAL SEARCHING AUTHORITY

Choice of International Searching Authority (ISA) *(if two or more International Searching Authorities are competent to carry out the international search, indicate the Authority chosen; the two-letter code may be used)*:

ISA / CN .....

Request to use results of earlier search; reference to that search *(if an earlier search has been carried out by or requested from the International Searching Authority)*:

Date *(day/month/year)* Number Country *(or regional Office)*

Box No. VIII DECLARATIONS

The following **declarations** are contained in Boxes Nos. VIII (i) to (v) *(mark the applicable check-boxes below and indicate in the right column the number of each type of declaration)*:		Number of declarations
☐ Box No. VIII (i)	Declaration as to the identity of the inventor :	
☐ Box No. VIII (ii)	Declaration as to the applicant's entitlement, as at the international filing date, to apply for and be granted a patent :	
☐ Box No. VIII (iii)	Declaration as to the applicant's entitlement, as at the international filing date, to claim the priority of the earlier application :	
☐ Box No. VIII (iv)	Declaration of inventorship (only for the purposes of the designation of the United States of America) :	
☐ Box No. VIII (v)	Declaration as to non-prejudicial disclosures or exceptions to lack of novelty :	

Box No. IX CHECK LIST; LANGUAGE OF FILING

This international application contains:

(a) **on paper**, the following number of sheets:

request (including declaration sheets)	:	3
description (excluding sequence listing and/or tables related thereto)	:	14
claims	:	5
abstract	:	1
drawings	:	9
Sub-total number of sheets	:	32
sequence listing	:	
tables related thereto	:	
(for both, actual number of sheets if filed on paper, whether or not also filed in electronic form; see (c) below)		
Total number of sheets	:	32

(b) ☐ **only in electronic form** (Section 801(a)(i))
- (i) ☐ sequence listing
- (ii) ☐ tables related thereto

(c) ☐ **also in electronic form** (Section 801(a)(ii))
- (i) ☐ sequence listing
- (ii) ☐ tables related thereto

Type and number of carriers (diskette, CD-ROM, CD-R or other) on which are contained the
- ☐ sequence listing: . . . . . .
- ☐ tables related thereto: . . . . . .

(additional copies to be indicated under items 9(ii) and/or 10(ii), in right column)

This international application is **accompanied by** the following item(s) *(mark the applicable check-boxes below and indicate in right column the number of each item)*:

		Number of items
1. ☒ fee calculation sheet	:	1
2. ☐ original separate power of attorney	:	
3. ☐ original general power of attorney	:	
4. ☐ copy of general power of attorney; reference number, if any:	:	
5. ☐ statement explaining lack of signature	:	
6. ☐ priority document(s) identified in Box No. VI as item(s):	:	
7. ☐ translation of international application into *(language)*:	:	
8. ☐ separate indications concerning deposited microorganism or other biological material	:	
9. ☐ sequence listing in electronic form *(indicate type and number of carriers)*		
(i) ☐ copy submitted for the purposes of international search under Rule 13*ter* only (and not as part of the international application)	:	
(ii) ☐ *(only where check-box (b)(i) or (c)(i) is marked in left column)* additional copies including, where applicable, the copy for the purposes of international search under Rule 13*ter*	:	
(iii) ☐ together with relevant statement as to the identity of the copy or copies with the sequence listing mentioned in left column	:	
10. ☐ tables in electronic form related to sequence listing *(indicate type and number of carriers)*		
(i) ☐ copy submitted for the purposes of international search under Section 802(b-*quater*) only (and not as part of the international application)	:	
(ii) ☐ *(only where check-box (b)(ii) or (c)(ii) is marked in left column)* additional copies including, where applicable, the copy for the purposes of international search under Section 802(b-*quater*)	:	
(iii) ☐ together with relevant statement as to the identity of the copy or copies with the tables mentioned in left column	:	
11. ☐ other *(specify)*:	:	

Figure of the drawings which should accompany the abstract:

Language of filing of the international application: English

Box No. X SIGNATURE OF APPLICANT, AGENT OR COMMON REPRESENTATIVE

Next to each signature, indicate the name of the person signing and the capacity in which the person signs (if such capacity is not obvious from reading the request).

北京英特普知识产权代理有限公司 IntellecPro

IntellecPro China Limited

For receiving Office use only

1. Date of actual receipt of the purported international application: 30 · 12月 2005 (30 · 12 · 2005)	2. Drawings: ☐ received:
3. Corrected date of actual receipt due to later but timely received papers or drawings completing the purported international application:	
4. Date of timely receipt of the required corrections under PCT Article 11(2):	☐ not received:
5. International Searching Authority (if two or more are competent): ISA /	6. ☐ Transmittal of search copy delayed until search fee is paid

For International Bureau use only

Date of receipt of the record copy by the International Bureau:

This sheet is not part of and does not count as a sheet of the international application.

PCT

FEE CALCULATION SHEET

Annex to the Request

For receiving Office use only

International Application No. PCT/CN2005 / 002386

Date stamp of the receiving Office 30·12月 2005 (30·12·2005)

Applicant's or agent's file reference: I05BJ1141

Applicant: INTEL CORPORATION, et al.

CALCULATION OF PRESCRIBED FEES

1. TRANSMITTAL FEE . . . . . CNY 500 [T] — CNY 500

2. SEARCH FEE . . . . . CNY 1500 [S] — CNY 1500

International search to be carried out by CN

(If two or more International Searching Authorities are competent to carry out the international search, indicate the name of the Authority which is chosen to carry out the international search.)

3. INTERNATIONAL FILING FEE

Where items (b) and/or (c) of Box No. IX apply, enter **Sub-total number of sheets** / Where items (b) and (c) of Box No. IX do not apply, enter **Total number of sheets**: 32

[i1] first 30 sheets . . . . . CHF 1400 [i1] — CHF 1400

[i2] 2 (number of sheets in excess of 30) x CHF 15 (fee per sheet) = CHF 30 [i2] — CHF 30

[i3] additional component (only if a sequence listing and/or tables related thereto are filed in electronic form under Section 801(a)(i), or both in that form and on paper, under Section 801(a)(ii)):

400 x _____ (fee per sheet) = _____ [i3]

Add amounts entered at i1, i2 and i3 and enter total at I . . . . CHF 1430 [I] — CHF 1430

(Applicants from certain States are entitled to a reduction of 75% of the international filing fee. Where the applicant is (or all applicants are) so entitled, the total to be entered at I is 25% of the international filing fee.)

4. FEE FOR PRIORITY DOCUMENT *(if applicable)* . . . . . [P]

5. TOTAL FEES PAYABLE . . . . . CHF 1430 CNY 2000 (TOTAL) — CNY 2000

Add amounts entered at T, S, I and P, and enter total in the TOTAL box

MODE OF PAYMENT *(Not all modes of payment may be available at all receiving Offices)*

- [x] authorization to charge deposit account (see below)
- [ ] postal money order
- [ ] cash
- [ ] coupons
- [ ] cheque
- [ ] bank draft
- [ ] revenue stamps
- [ ] other *(specify)*:

AUTHORIZATION TO CHARGE (OR CREDIT) DEPOSIT ACCOUNT

(This mode of payment may not be available at all receiving Offices)

Receiving Office: RO/ CN

- [x] Authorization to charge the total fees indicated above.
- [ ] *(This check-box may be marked only if the conditions for deposit accounts of the receiving Office so permit)* Authorization to charge any deficiency or credit any overpayment in the total fees indicated above.
- [ ] Authorization to charge the fee for priority document.

Deposit Account No.: _____

Date: _____

Name: _____

Signature: _____

See Notes to the fee calculation sheet

VIRTUAL MACHINE TO DETECT MALICIOUS CODE

FIELD

[0001] Embodiments of the invention relates to virtual machines, and more particularly to detection of malicious code by a virtual machine.

BACKGROUND

[0002] Computer networking is prevalent amongst many users of computing devices, such as personal computers and workstations. Networking allows users of computing devices to communicate with each other in various forms, such as the exchange of data or computer programs which can be downloaded from the network and run on each computing device. A typical network environment, however, includes computing devices which operate on different (and often incompatible) operating systems host platforms, such as Windows®, DOS™, Linux®, etc, thus making it difficult for a downloaded computer program to be directly run on the different computing devices.

[0003] One prevalent approach to the foregoing problem is by the use of virtual machine, in a computing device. A virtual machine, such as dynamic binary translator, Just-in-Time compiler, or Java Virtual Machine Interpreter, etc. is an abstract computing device that virtualizes an environment on which a computer program can run on a host platform. In this way, the same computer program can be run on different (and otherwise incompatible) operating systems host platforms. In addition a virtual machine can enable a computer program to run on computers with different architectures.

[0004] The use of virtual machines, while effective for running computer programs on different operating systems host platforms, is not without shortcomings in other respects, such as in the area of security. The security issues arise from the added vulnerability of a computing device to malicious code while using the virtual machine. Malicious code, also termed as malware, describes the code fragments intentionally performing an unauthorized process, and which can

invade a computing device across the network. Variants of malicious code are virus, worm, Trojan horse, spyware, adware, logic bomb and backdoors. Generally, virtual machines prevent the traditional anti-malware software, which are individual programs, from catching the malicious code running on top of them or the host platform, because in such situations the anti-malware software would not be effective without support from the virtual machine.

[0005] One situation in which anti-malware software would not be effective is when the individual anti-malware software runs on top of the host platform. The anti-malware software will then fail to emulate the monitored program's execution before the monitored program really starts. This emulation is necessary to modern anti-malware software because of the emergence of polymorphism viruses. The polymorphism viruses self-encrypt with different decryption routines to produce varied but operational copies of themselves, so polymorphism viruses don't have fixed code patterns in the executable image file. To detect them, the anti-malware software must run the monitored program in an emulated and insulated environment before the program actually starts. During the emulation, the anti-malware software scans virus signatures in the emulated memory. For performance considerations, however, if after a period of time the virus signatures have not been found, the emulation stops and the monitored program then starts. Since the target host platform is determined to the anti-malware software, the anti-malware software prepares a simulator for the host platform before hand. But predicting which virtual machines are going to be installed on the host platform is difficult, thus making it impractical for the individual anti-malware software to prepare simulators for all virtual machines beforehand. In addition, simulating a virtual machine will be too complex to the individual anti-malware software, which degrades the performance to unacceptable levels.

[0006] In addition, if the individual anti-malware software runs on top of the host platform, it will fail to intercept the original system calls issued from

the interpreter functions and translation cache of virtual machine environment. In this scenario, some anti-malware software intercepts system calls from the monitored program to detect malicious code. The system calls issued from interpreter functions and translation cache, however, were converted by the system call converter before the anti-malware software intercepts them, which will mislead the anti-malware software. Moreover, the individual anti-malware software typically fails to run on most virtual machines because privileged instructions are included in individual anti-malware software but are not supported by most virtual machines.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] Embodiments of the invention may best be understood by referring to the following description and accompanying drawings that are used to illustrate embodiments of the invention.

[0008] **FIG. 1** is an exemplary block diagram of computing device in which embodiments of the invention may be practiced.

[0009] **FIGs. 2A-6** are exemplary flow charts illustrating processes according to an exemplary embodiment of the invention.

[00010] **FIGs. 7A-B** are exemplary flow charts illustrating processes according to another exemplary embodiment of the invention.

[00011] **FIGs. 8A-B** are exemplary flow charts illustrating processes according to yet another exemplary embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

[00012] Embodiments of the invention generally relate to systems and methods for detection of malicious code by a virtual machine. Herein, embodiments of the invention may be applicable to virtual machines used in a variety of computing devices, which are generally considered stationary or portable electronic devices. Examples of computing devices include any type of stationary or portable electronic device that may be adversely effected by malware such as a computer, work station, a set-top box, a wireless telephone, a digital video recorder (DVR), networking equipment (e.g., routers, servers, etc.) and the like.

[00013] Certain details are set forth below in order to provide a thorough understanding of various embodiments of the invention, albeit embodiments of the invention may be practiced through many embodiments of the invention other than those illustrated. Well-known logic and operations are not set forth in detail in order to avoid unnecessarily obscuring this description.

[00014] In the following description, certain terminology is used to describe features of the various embodiments of the invention. The term "software" generally denotes executable code such as an operating system, an application, an applet, a routine or even one or more instructions. The software may be stored in any type of memory, namely suitable storage medium such as a programmable electronic circuit, a semiconductor memory device, a volatile memory (e.g., random access memory, etc.), a non-volatile memory (e.g., read-only memory, flash memory, etc.), a floppy diskette, an optical disk (e.g., compact disk or digital versatile disc "DVD"), a hard drive disk, tape, or any kind of interconnect.

[00015] In general terms, a virtual machine (also known as software dynamic translator) creates an environment between a host platform on a computer and an end-user, in which the end user can operate software otherwise incompatible with the host platform. Variants of virtual machine are dynamic binary translator, interpreters, and just-in-time (JIT) compilers. A "host platfrom" is an operating system, such as Windows®, DOS™ and Linux®, which enables a computing

device to run various softwares. A malicious code, also termed as "malware", describes the code fragments intentionally performing unauthorized tasks. Variants of malicious code are virus, worm, Trojan horse, spyware, adware, logic bomb and backdoors. A "translation cache" describes reusable translated code generated by a virtual machine that is unnecessary to exist in processor. An "interpreter" is a program that executes other programs, such as a Java Interpreter executing Java® programs.

[00016] With reference to **FIG. 1**, a block diagram of a computing device 100 in which embodiments of the invention may be practiced is shown. As shown in FIG. 1, the computing device 100 includes a virtual machine 120 to receive contents 112 of a source program 113, such as instructions 114 and metadata 115, for creating a virtual environment for interacting with a host platform 110 in the computing device 100. The virtual machine 120 comprises a detection subsystem 101 to determine if the received contents 112 comprises predetermined instructions for performing unauthorized tasks, such as malware instructions as defined above. The virtual machine 120 is then to purge the predetermined instructions from at least one of the source program 113 or the received contents 112 of the source program 113.

[00017] The detection subsystem 101 further comprises a comparator logic 117 to compare the received contents 112 to at least one predetermined instruction pattern stored in a detection database 116, which corresponds to the predetermined instructions for performing unauthorized tasks. The detection database 116 may be external to the detection subsystem 101 or the virtual machine 120. Suitably, the comparator logic 117 includes a search logic (not shown) to first search predetermined locations of the contents 112 for the predetermined instructions for performing unauthorized tasks, as described below and in greater detail in conjunction with FIG. 3. The comparator logic 117 may be implemented in hardware or software stored on a memory storage medium (not shown).

[00018] As also shown in FIG. 1, the virtual machine 120 further comprises at least one translation cache 104, such as translation cache_1 through translation cache_N (N≥1). The virtual machine 120 also includes a translation engine 103 to invoke the detection subsystem 101 to determine if the instructions 114 in the source program 113 comprises predetermined instructions for performing unauthorized tasks, as described below and in greater detail in conjunction with FIG. 4. The virtual machine 120 also includes a loader 111 to receive contents 112 of the source program 113 and to invoke the detection subsystem 101.

[00019] The virtual machine 120 may also include interpreter functions 105, such as function_1 through function _M (M>1), and an execution engine 102 to invoke the detection subsystem 101 to determine if the instructions 114 in the source program 113 that may include predetermined instructions for performing unauthorized tasks prior to invoking the interpreter functions 105, as described below and in greater detail in conjunction with FIG. 6. It should be noted that interpreter functions 105, translation engine 103 and translation cache 104 are implementation dependent, so that an exemplary virtual machine 120 may have only the interpreter function 105 feature (such as in a Java interpreter implementation), or only the translation engine 103 and translation cache 104 feature (such as in a Java Just-In-Time (JIT) compiler implementation), or both features. Typically, an interpreter function 105 simulates an instruction from the source program 113 and is prepared at the build time, whereas a piece of translation cache 104 is able to simulate a number of instructions and is generated by the translation engine 103 at runtime.

[00020] Interpreter functions 105 and translation cache 104 use the services provided by the address converter 106 and system call converter 109. The address converter 106 converts received virtualized memory addresses, which are used by interpreter functions 105 and translation cache 104, into memory addresses meaningful to the host platform 110 before the memory accesses really happens. The system call converter 109 converts system calls issued from interpreter functions 105 and translation cache 104 into the meaningful system calls to the host

platform 110. In an embodiment of the invention, a system call filter 108 is implemented to filter out system calls for performing unauthorized tasks, as described below and in greater detail in conjunction with FIGs. 7A-7B. Suitably, communications between the execution engine 102, translation engine 103, translation cache 104, interpreter functions 105, address converter 106, system call filter 108, loader 111, detection subsystem 101, as well as other components (not shown) of the computing device 100 are enabled via a bus 107.

[00021] **FIG. 2A** is an exemplary flow chart illustrating a process according to an exemplary embodiment of the invention. As shown in FIG. 2A (in conjunction with FIG. 1), following the start of the process (block 200), contents 112 of a program are received in the virtual machines 120 for creating a virtual environment for interacting with host platform 110 in the computing device 100 (block 210), as described below and in greater detail in conjunction with FIG. 2B. The virtual machine 120 then determines if the received contents 112 comprises predetermined instructions for performing unauthorized tasks (block 220), as described below and in greater detail in conjunction with FIG. 2B. The overall process then ends (block 230).

[00022] **FIG. 2B** is an exemplary flow chart illustrating the operations of FIG. 2A in conjunction with FIG. 1. As shown in FIG. 2B, following the start of the process (block 200), contents 112 of a program are received in the virtual machines 120 (block 210). In an exemplary embodiment of the invention, the contents 112 are first received in the loader 111 from the source program 113 (block 240), as shown symbolically by line 112 in FIG. 1. Then, the loaded contents 112 are examined for detection of any predetermined instructions for performing unauthorized tasks, such as malicious code (block 250), as described below and in greater detail in conjunction with FIG. 3. If malicious code is detected, attempts are made to purge the malicious code (block 255). If the malicious code cannot be purged, then the overall process ends (block 230). If the no malicious code is detected, or if the detected malicious code is successfully purged (blocks 250, 255), an instruction pointer (IP) is initialized to point to an

instruction in the loaded contents 112 to be executed (block 260), such as initializing IP to point to the first instruction in the loaded contents 112.

[00023] The process then involves a determination of whether the instruction address in IP resides in available address space (block 265). If the instruction address in IP does not reside in available address space, the overall process ends (block 230). Otherwise, it is determined if the virtual machine 120 uses translation cache 104, such as when the virtual machine 120 includes a Java JIT complier (block 270). Next, prior to generating translation cache 104, the instructions 114 in the source program 113 are tested again to determine if they may include malicious code (block 275), as described below and in greater detail in conjunction with FIGs. 4-5. If it is determined the virtual machine 120 does not use translation cache 104, or following completion of detection of malicious code in translation cache 104 (blocks 270, 275), then it is determined if the virtual machine 120 uses interpreter functions 105 of FIG. 1, such as when the virtual machine 120 includes a Java interpreter (block 280). If so, prior to invoking the interpreter functions 105, the instructions 114 in the source program 113 are tested to determine if they may include malicious code (block 285), as described below and in greater detail in conjunction with FIG. 6. Following the operations of block 285, or if the virtual machine 120 uses interpreter functions 105 (block 280), it is determined if more instruction are to be executed (block 290). If so, further processing continues (block 265), and if not, the overall process ends (block 230).

[00024] **FIG. 3** is an exemplary flow chart which, in conjunction with FIG. 1, further illustrates the detection and purging of the malicious code shown in FIG. 2B (blocks 250, 255). As shown in FIG. 3, following the start of the process (block 300), predetermined locations of the received contents 112 of the program 113 are searched for the malicious code (block 310). In an exemplary embodiment of the invention, the loader 111 invokes the detector subsystem 101, as shown symbolically by line 20 in FIG. 1, for performing the search. The detection subsystem 101 comprises a detection database 116 which contains possible locations in received contents 112 in which malicious code may reside. Next, the

contents 112 of the program 113 are compared to predetermined instruction patterns corresponding to the malicious codes which perform unauthorized tasks (block 320). In an exemplary embodiment of the invention, the comparison is performed by the comparator 117 in communication with the detection database 116, which contains predetermined instruction patterns corresponding to the malicious codes. If a match is found, then a malicious code is deemed detected. Following the comparing, the malicious code is purged from the received contents 112 by the detector subsystem (block 330). Suitably, the detection database 116 contains a prescription on how to purge the malicious code from the received contents 112. The flow is then returned to block 255 of FIG. 2B (block 340).

[00025] **FIG. 4** is an exemplary flow chart which in conjunction with FIG. 1 further illustrates the detection of the malicious code in the instructions 114 of the source program 113 shown in FIG. 2B (block 275). As shown in FIG. 4, following the start of the process (block 400), it is determined if a translation cache 104, such as translation cache_1, corresponding to the value in the instruction pointer (IP) (initialized in block 260 of FIG. 2B) exists (block 410). If so, no malicious code is detected and the flow is returned to block 275 of FIG. 2B (block 495), otherwise, the translation engine 103 is invoked by the execution engine 102 (block 420), as shown symbolically by line 17 in FIG. 1. The translation engine 103 then invokes the detection subsystem 101 (block 430), as shown symbolically by line 18 in FIG. 1.

[00026] Next, starting from the instruction that IP points to, the translation engine 103 traverses code fragments in the instructions 114 in the source program 113 (block 440). For each traversed code fragment, the translation engine 103 invokes the detection subsystem 101 to compare the traversed code with the code patterns of malicious code (block 450). If no match is found, then no malicious code is detected and the flow is returned to block 275 of FIG. 2B (blocks 460, 495). If a match is found, malicious code is detected (block 460), in which case the virtual machine 120 attempts to purge the malicious code from the traversed code fragment by following the prescription in the record stored in the detection database 116 (block 470). If the purge was unsuccessful the flow is returned to block 275 of FIG. 2B (blocks 475,

495), the execution operations of the virtual machine 120 is stopped for the loaded contents 112. If the purge was successful, it is determined if more code fragments are to be traversed (block 480) and if so, the process is returned to block 440, otherwise the translation engine 103 generates a translation cache 104, such as translation cache_2, for the traversed code fragments (block 485), as shown symbolically by line 13 in FIG. 1. The execution engine 102 then directs control to the translation cache 104 corresponding to the IP, such as to translation cache_2 (block 490), as shown symbolically by line 16 in FIG. 1.

[00027] When the control reaches an outlet of a translation cache 104, the IP has been updated and the translation cache 104 should direct the control back to the execution engine 102, as shown symbolically by line 16 in FIG. 1, or to another translation cache 104. In an exemplary embodiment of the invention, before the control is actually directed back to the execution engine 102 or to another translation cache 104, additional safety measures are undertaken to reduce the occurrence of malicious code directing the control to an unauthorized location, as described below and in greater detail in conjunction with FIG. 5.

[00028] **FIG. 5** is an exemplary flow chart which in conjunction with FIG. 1 further illustrates the detection of the malicious code in the instructions 114 in the source program 113 prior to generating of the translation cache 104 shown in FIG. 2B (block 275). As shown in FIG.5, following the start of the process (block 500), a branch target at the outlets of a translation cache 104 is checked (block 520). If the branch target is not a piece of either translation cache 104 or the execution engine 102 (blocks 540, 550), then malicious code is deemed detected (block 560). The control is then directed back to the execution engine 102 (block 570), as shown symbolically by line 16 in FIG. 1, which then stops the execution operations of the virtual machine 120 for the loaded contents 112, following the return of the flow to block 275 of FIG. 2B (block 580). If the branch target is a piece of either translation cache 104 or the execution engine 102 (blocks 540, 550), then malicious code is deemed not detected and the flow is returned to block 275 of FIG. 2B (block 580). Suitably, prior to the operations of FIG. 5, the

translation engine 103 generates translation cache logic instructions for performing the foregoing operations described in conjunction with FIG. 5.

[00029] **FIG. 6** is an exemplary flow chart which in conjunction with FIG. 1 further illustrates the detection of the malicious code in the instructions 114 in the source program 113 prior to invoking of the interpreter functions 105 shown in FIG. 2B (block 285). As shown in FIG. 6, following the start of the process (block 600), the execution engine 102 invokes the detection subsystem 101, as shown symbolically by line 19 in FIG. 1. Next, starting from the instruction that IP points to, the execution engine 102 traverses code fragments instructions 114 in the source program 113 (block 620). For each traversed code fragment, the invoked detection subsystem 101 compares the traversed code with the code patterns of malicious code (block 630). If no match is found, then no malicious code is detected and the flow is returned to block 285 of FIG. 2B (blocks 640, 699). If a match is found, malicious code is detected (block 640), in which case the virtual machine 120 attempts to purge the malicious code from the traversed code fragment by following the prescription in the record stored in the detection database 116 (block 650). If the purge was unsuccessful the flow is returned to block 285 of FIG. 2B (blocks 660, 699), and the execution operations of the virtual machine 120 are stopped for the loaded contents 112. If the purge was successful, it is determined if more code fragments are to be traversed (block 670), and if so, the process is returned to block 620. Otherwise the execution engine 102 decodes the instructions that IP points to (block 680).

[00030] Next, the execution engine 102 directs the control to the corresponding interpreter function 105, such as to function_2, as shown symbolically by line 12 in FIG. 1. Upon completion of the execution by the interpreter functions 105, the control is directed back to the execution engine 102 with an updated IP (block 695), as shown symbolically by line 12 in FIG. 1. The flow is then returned to block 285 of FIG. 2B (block 699).

[00031] **FIGs. 7A-B** are exemplary flow charts illustrating processes according to another exemplary embodiment of the invention. As described above in conjunction with FIG. 1, interpreter functions 105 and translation cache 104 use the services provided by the system call converter 109. The system call converter 109 converts system calls issued from interpreter functions 105 and translation cache 104 into the meaningful system calls to the host platform 110. In an exemplary embodiment of the invention, a system call filter 108 is implemented to filter out system calls for performing unauthorized tasks. Exemplary operations of the system call filter 108 is described in conjunction with FIGs. 7A-B.

[00032] As shown in **FIG. 7A**, following the start of the process (block 700), a system call for the host platform 110 is received in the system call filter 108 (block 710), such as via a system call interception. The virtual machine 120 then determines if the received system call contains predetermined system calls for performing unauthorized tasks (block 720), as described in greater detail in conjunction with FIG. 7B below. The overall flow then ends (block 730).

[00033] **FIG. 7B** is an exemplary flow chart which in conjunction with FIG. 1 further illustrates the operations shown in FIG. 7A (block 720) to determine if the received system call comprises predetermined system calls for performing unauthorized tasks. As shown in FIG. 7B, following the start of the process (block 750), the received system call is compared to predetermined system calls patterns corresponding to the predetermined system calls for performing unauthorized tasks (block 760). In an exemplary embodiment of the invention, a system call is determined to be for performing unauthorized tasks if its task is inhibitive, or results in outputting of data into the memory regions storing instructions or data for operations of the virtual machine 120 and its components, including the translation cache 104. If the system call is determined to be unauthorized, malicious code is deemed detected and the operations of the virtual machine 120 corresponding to the system call will be stopped. Otherwise, if the system call is determined to be authorized, the system call filter 108 passes the system call to system call converter 109. The flow is then returned to block 720 of FIG. 7A (block 770).

[00034] **FIGs. 8A-B** are exemplary flow charts illustrating processes according to yet another exemplary embodiment of the invention. As described above in conjunction with FIG. 1, interpreter functions 105 and translation cache 104 use the services provided by the address converter 106. The address converter 106 converts received virtualized memory addresses, which are used by interpreter functions 105 and translation cache 104, into memory addresses meaningful to the host platform 110 before the memory accesses really happens. In an exemplary embodiment of the invention, before the address converter 106 converts a received virtualized memory address to a memory address meaningful to the host platform 110, it checks the received virtualized memory address to determines if the received virtualized memory address is an unauthorized virtualized memory address, as described in greater detail in conjunction with FIGs. 8A-B.

[00035] As shown in **FIG. 8A**, following the start of the process (block 800), a virtualized memory address for the host platform 110 is received in the address converter 106 (block 810). The virtual machine 120 then determines if the received virtualized memory address comprises predetermined unauthorized virtualized memory address (block 820), as described in greater detail in conjunction with FIG. 8B below. The overall flow then ends (block 830).

[00036] **FIG. 8B** is an exemplary flow chart which in conjunction with FIG. 1 further illustrates the operations shown in FIG. 8A (block 820) to determine if the received virtualized memory address comprises predetermined unauthorized virtualized memory address. As shown in FIG. 8B, following the start of the process (block 850), it is determined if the received virtualized memory address is in a memory space available to a) the translation cache 104 (block 860), or b) to an interpret function 105(block 870), or if c) the virtualized memory address is in a memory space region storing instructions or data for operations of the virtual machine 120 (block 880). If so, malicious code is deemed detected and the operation of the virtual machine 120 utilizing the received virtualized memory address is stopped, otherwise the address converter 106 converts the received

virtualized memory address to a memory address meaningful to the host platform 110. The flow is then returned to block 820 of FIG. 8A (block 890).

[00037] In an exemplary embodiment of the invention, the software that, if executed by a computing device 100, will cause the computing device 100 to perform the above operations described in conjunction with FIGs. 2-8B is stored in a storage medium (not shown), such as main memory, or other storage devices such as a hard-disk.

[00038] It should be noted that the various features of the foregoing embodiments of the invention were discussed separately for clarity of description only and they can be incorporated in whole or in part into a single embodiment of the invention having all or some of these features.

CLAIMS

What is claimed is:

1. A method comprising:

receiving in a virtual machine contents of a program for creating a virtual environment for interacting with a host platform in a computing device; and

determining by the virtual machine if the received contents comprises predetermined instructions for performing at least one unauthorized task.

2. The method of claim 1, wherein the determining if the received contents comprises predetermined instructions further comprises:

comparing the received contents of the program to at least one predetermined instruction patterns corresponding to the predetermined instructions for performing the at least one unauthorized task; and

purging the predetermined instructions from the received contents based on the comparing.

3. The method of claim 2, wherein the comparing the contents of the received program to at least one predetermined instruction patterns further comprises:

searching predetermined locations of the received contents of the program for the predetermined instructions.

4. The method of claim 2, wherein the virtual machine comprises a translation cache, wherein the contents of the program reside in the translation cache, and wherein determining if the received contents comprises predetermined instructions further comprises:

checking a branch target at the outlets of the translation cache; and

determining if the checked branch target comprises at least one of a translation cache and the execution engine.

5. The method of claim 4, further comprising:

generating checking and determining instructions for performing the checking the branch target and determining if the checked branch target comprises at least one of a translation cache and the execution engine.

6. The method of claim 2, wherein the virtual machine comprises an execution engine and at least one interpret function invoked by the execution engine, wherein the contents of the program reside in the at least one interpret function.

7. A system comprising:

a virtual machine to receive contents of a program for creating a virtual environment for interacting with a host platform in a computing device, the virtual machine comprising a detector subsystem to determine if the received contents comprises predetermined instructions for performing at least one unauthorized task.

8. The system of claim 7, wherein the detector subsystem is to purge the predetermined instructions from the received contents of the program, wherein the detector subsystem further comprises:

a comparator logic to compare the received contents of the program to at least one predetermined instruction patterns corresponding to the predetermined instructions for performing the at least one unauthorized task; and

a search logic to search predetermined locations of the received contents of the program for the predetermined instructions.

9. The system of claim 7, wherein the virtual machine comprises:

at least one of a translation cache to store translation data;

a translation engine to invoke the detector subsystem to determine if the contents of a translation data storage comprises predetermined instructions for performing at least one unauthorized task;

at least one loader, to receive contents of a program and to invoke the detector subsystem;

at least one interpreter function; and

an execution engine to invoke the detector subsystem to determine if the contents of the at least one interpret function invoked by the execution engine comprises predetermined instructions for performing at least one unauthorized task.

10. The system of claim 9, wherein the detector subsystem further comprises:

translation cache logic to check a branch target at the outlets of the translation cache and to determine if the checked branch target comprises at least one of a translation cache and the execution engine, based on translation cache logic instructions; and

an instruction generation subsystem to generate the translation cache logic instructions.

11. The method of claim 8, wherein the at least one predetermined instruction patterns are stored in a database in communication with the virtual machine.

12. A storage medium that provides software that, if executed by a computing device, will cause the computing device to perform the following operations:

receiving in a virtual machine contents of a program for creating a virtual environment for interacting with a host platform in a computing device; and

determining by the virtual machine if the received contents comprises predetermined instructions for performing at least one unauthorized task.

13. The storage medium of claim 12 further comprising software to:

compare the received contents of the program to at least one predetermined instruction patterns corresponding to the predetermined instructions for performing the at least one unauthorized task; and

purge the predetermined instructions from the received contents based on the comparing.

14. The storage medium of claim 13 further comprising software to:

search predetermined locations of the received contents of the program for the predetermined instructions

15. A method comprising:

receiving a system call for a host platform in communication with a virtual machine of a computing device; and

determining by the virtual machine if the received system call comprises at least one predetermined system call for performing at least one unauthorized task.

16. The method of claim 15, wherein the determining if the received system call comprises predetermined system call further comprises:

comparing the system call to at least one predetermined system call patterns corresponding to the predetermined system calls for performing the at least one unauthorized task.

17. The method of claim 16, wherein the unauthorized task comprises:

a task predetermined to be an inhibitive task by the computing device; and

a task to output data into memory regions storing at least one of instructions and data for operations of the virtual machine.

18. A method comprising:

receiving a virtualized memory address for a host platform in communication with a virtual machine of a computing device; and

determining by the virtual machine if the received virtualized memory address comprises at least one predetermined unauthorized virtualized memory address.

19. The method of claim 18, wherein the virtual machine further comprises:

at least one of a translation cache to store translation data;

an execution engine; and

at least one interpret function invoked by the execution engine.

20. The method of claim 19, wherein the determining by the virtual machine if the received virtualized memory address comprises at least one predetermined unauthorized virtualized memory address comprises:

determining if the virtualized memory address is in a memory space available to the translation cache;

determining if the virtualized memory address is in a memory space available to the at least one interpret function; and

determining if the virtualized memory address is in a memory space region storing at least one of instructions and data for operations of the virtual machine.

ABSTRACT

One embodiment of the invention discloses a method for receiving in a virtual machine(VM) contents of a program for creating a virtual environment for interacting with a host platform in a computing device; and determining by the VM if the received contents comprise predetermined instructions for performing at least one unauthorized task. Another embodiment of the invention discloses a method for receiving a system call for a host platform in communication with a VM of a computing device; and determining by the VM if the received system call comprises at least one predetermined system call for performing at least one unauthorized task. Yet another embodiment of the invention discloses a method for receiving a virtualized memory address for a host platform in communication with a VM of a computing device; and determining by the VM if the received virtualized memory address comprises at least one predetermined unauthorized virtualized memory address.

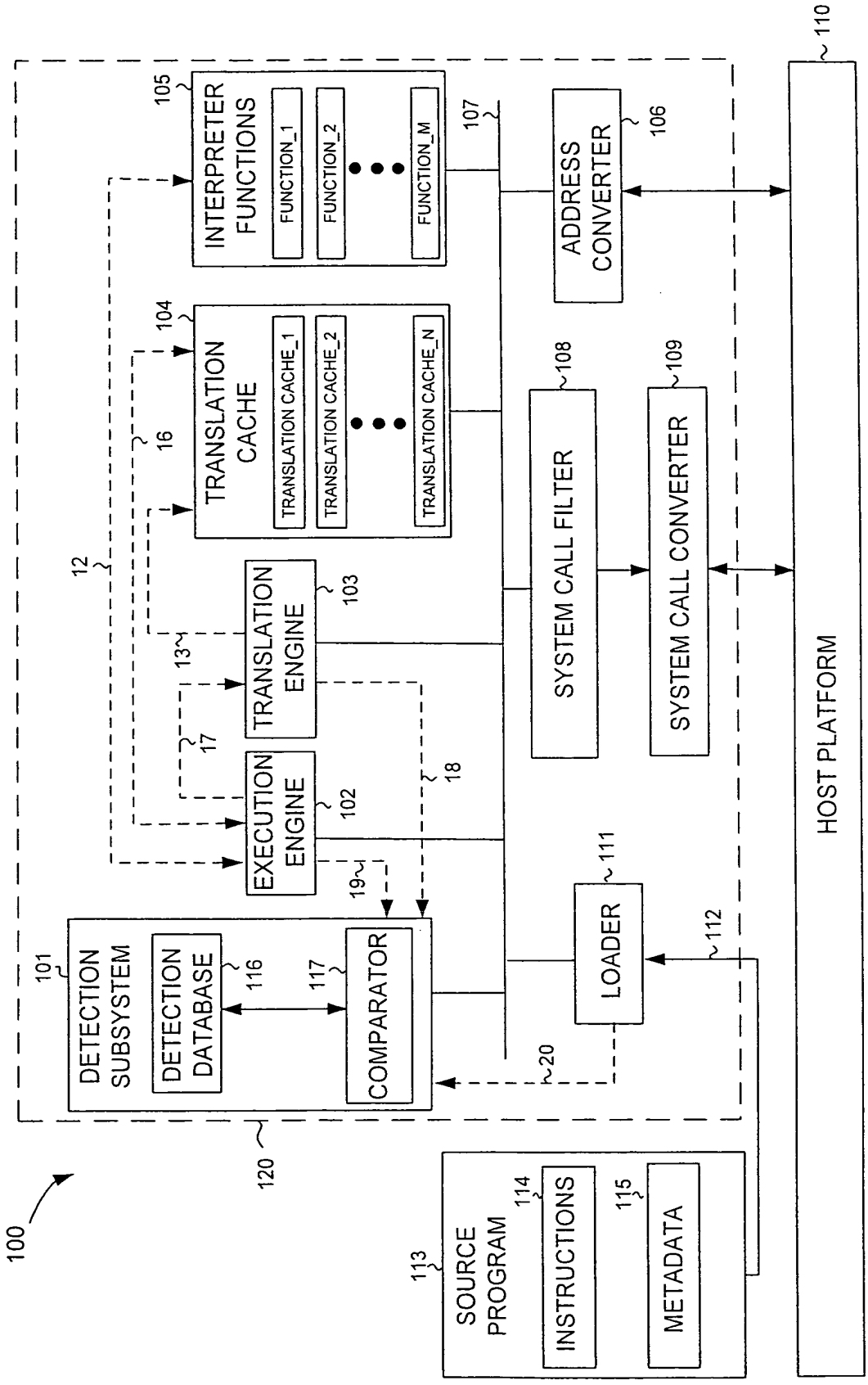
100
120
DETECTION SUBSYSTEM
101
DETECTION DATABASE
116
117
COMPARATOR
EXECUTION ENGINE
102
TRANSLATION ENGINE
103
12
13
16
17
18
19
20
TRANSLATION CACHE
104
TRANSLATION CACHE_1
TRANSLATION CACHE_2
TRANSLATION CACHE_N
INTERPRETER FUNCTIONS
105
FUNCTION_1
FUNCTION_2
FUNCTION_M
107
ADDRESS CONVERTER
106
SYSTEM CALL FILTER
108
SYSTEM CALL CONVERTER
109
HOST PLATFORM
110
LOADER
111
112
SOURCE PROGRAM
113
INSTRUCTIONS
114
METADATA
115

FIG. 1

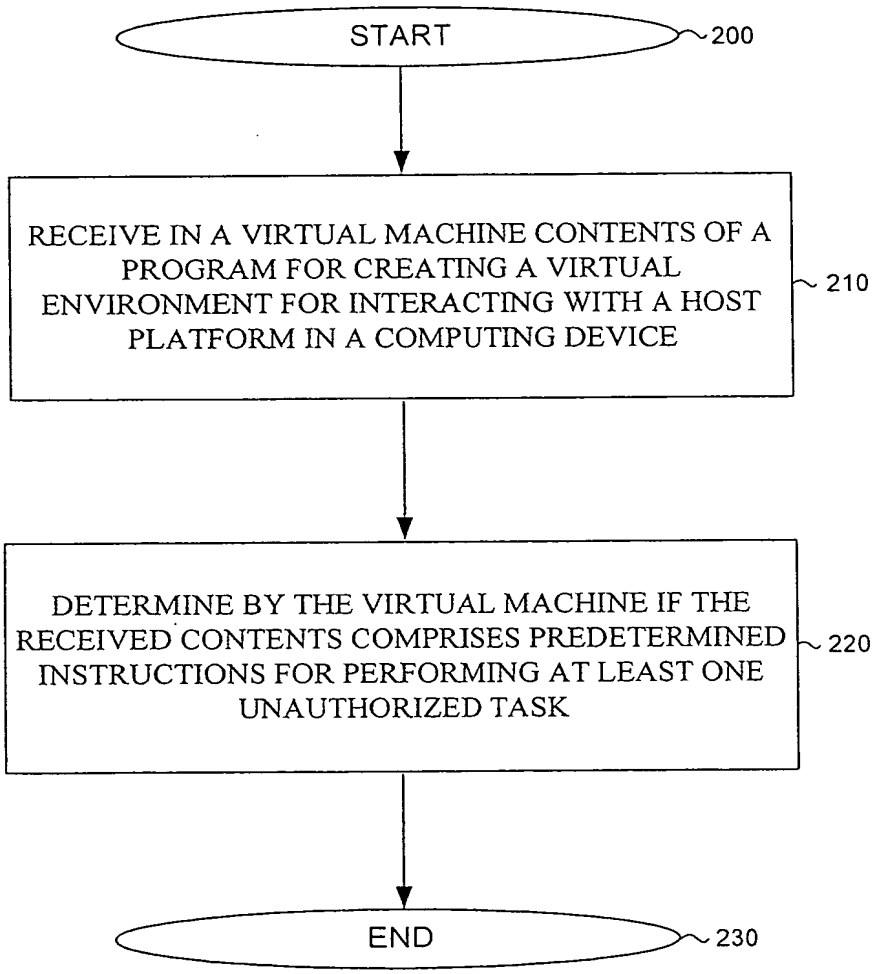
START
200
RECEIVE IN A VIRTUAL MACHINE CONTENTS OF A PROGRAM FOR CREATING A VIRTUAL ENVIRONMENT FOR INTERACTING WITH A HOST PLATFORM IN A COMPUTING DEVICE
210
DETERMINE BY THE VIRTUAL MACHINE IF THE RECEIVED CONTENTS COMPRISES PREDETERMINED INSTRUCTIONS FOR PERFORMING AT LEAST ONE UNAUTHORIZED TASK
220
END
230

FIG. 2A

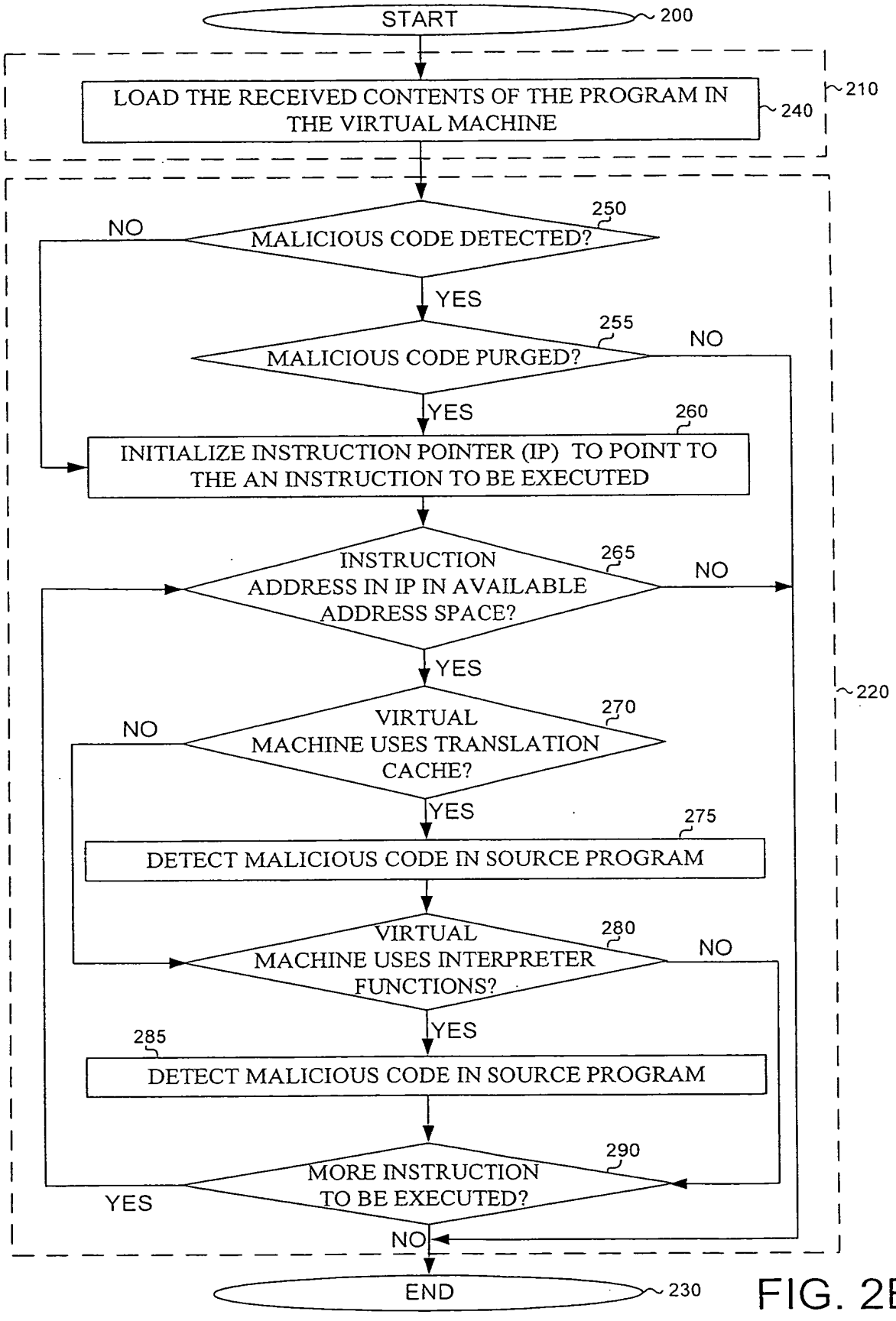
START
200
LOAD THE RECEIVED CONTENTS OF THE PROGRAM IN THE VIRTUAL MACHINE
240
210
MALICIOUS CODE DETECTED?
250
NO
YES
MALICIOUS CODE PURGED?
255
NO
YES
INITIALIZE INSTRUCTION POINTER (IP) TO POINT TO THE AN INSTRUCTION TO BE EXECUTED
260
INSTRUCTION ADDRESS IN IP IN AVAILABLE ADDRESS SPACE?
265
NO
YES
VIRTUAL MACHINE USES TRANSLATION CACHE?
270
NO
YES
DETECT MALICIOUS CODE IN SOURCE PROGRAM
275
VIRTUAL MACHINE USES INTERPRETER FUNCTIONS?
280
NO
YES
DETECT MALICIOUS CODE IN SOURCE PROGRAM
285
MORE INSTRUCTION TO BE EXECUTED?
290
YES
NO
220
END
230

FIG. 2B

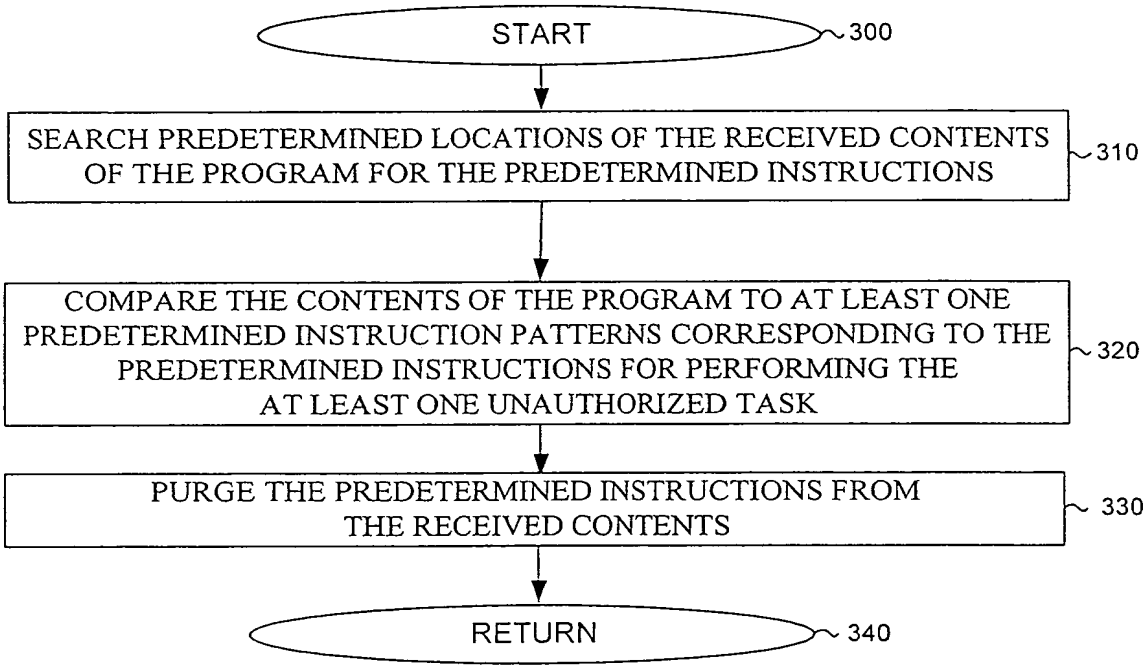
START
300
SEARCH PREDETERMINED LOCATIONS OF THE RECEIVED CONTENTS OF THE PROGRAM FOR THE PREDETERMINED INSTRUCTIONS
310
COMPARE THE CONTENTS OF THE PROGRAM TO AT LEAST ONE PREDETERMINED INSTRUCTION PATTERNS CORRESPONDING TO THE PREDETERMINED INSTRUCTIONS FOR PERFORMING THE AT LEAST ONE UNAUTHORIZED TASK
320
PURGE THE PREDETERMINED INSTRUCTIONS FROM THE RECEIVED CONTENTS
330
RETURN
340

FIG. 3

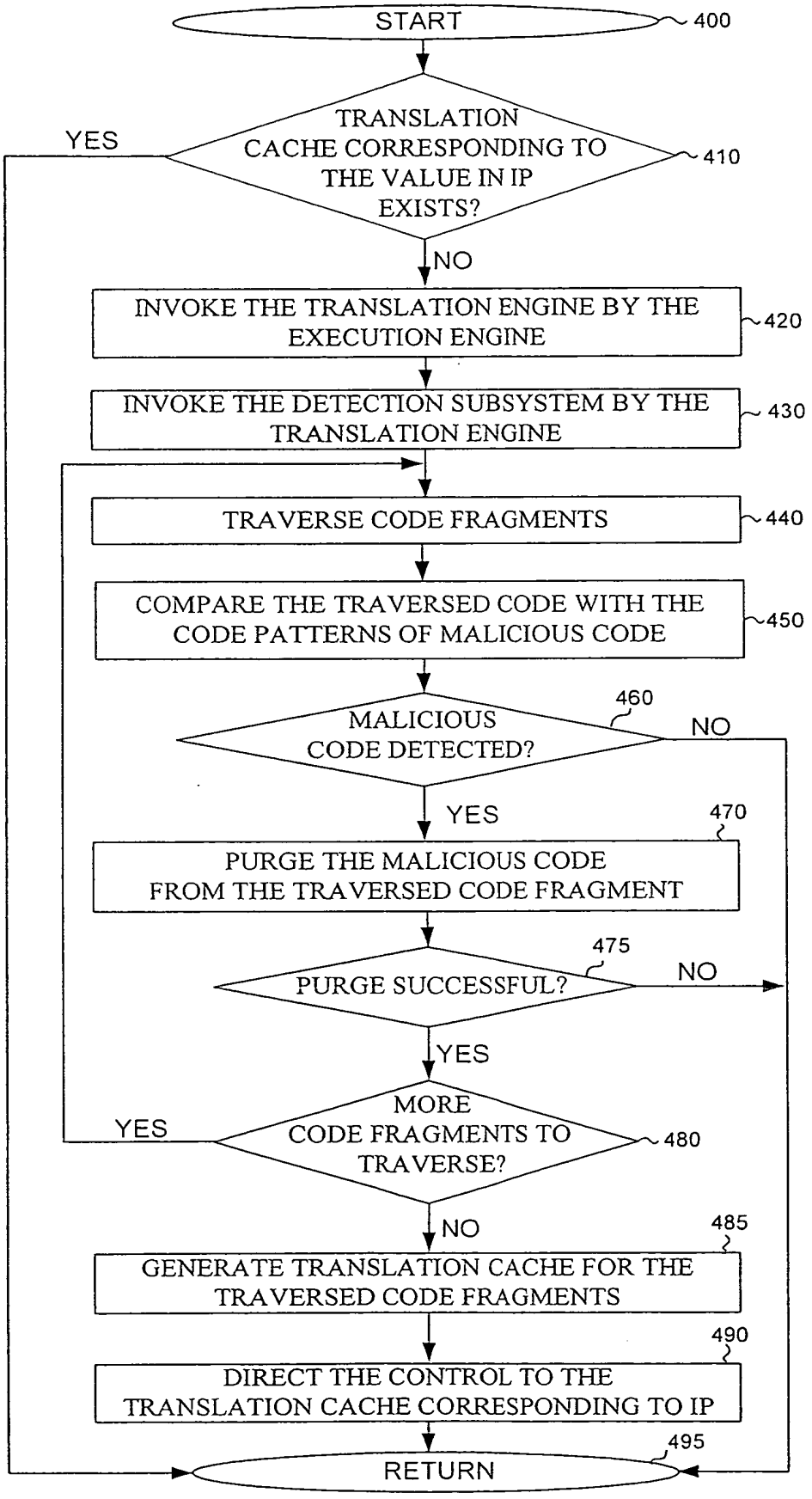
START
400
TRANSLATION CACHE CORRESPONDING TO THE VALUE IN IP EXISTS?
410
YES
NO
INVOKE THE TRANSLATION ENGINE BY THE EXECUTION ENGINE
420
INVOKE THE DETECTION SUBSYSTEM BY THE TRANSLATION ENGINE
430
TRAVERSE CODE FRAGMENTS
440
COMPARE THE TRAVERSED CODE WITH THE CODE PATTERNS OF MALICIOUS CODE
450
MALICIOUS CODE DETECTED?
460
NO
YES
PURGE THE MALICIOUS CODE FROM THE TRAVERSED CODE FRAGMENT
470
PURGE SUCCESSFUL?
475
NO
YES
MORE CODE FRAGMENTS TO TRAVERSE?
480
YES
NO
GENERATE TRANSLATION CACHE FOR THE TRAVERSED CODE FRAGMENTS
485
DIRECT THE CONTROL TO THE TRANSLATION CACHE CORRESPONDING TO IP
490
RETURN
495

FIG. 4

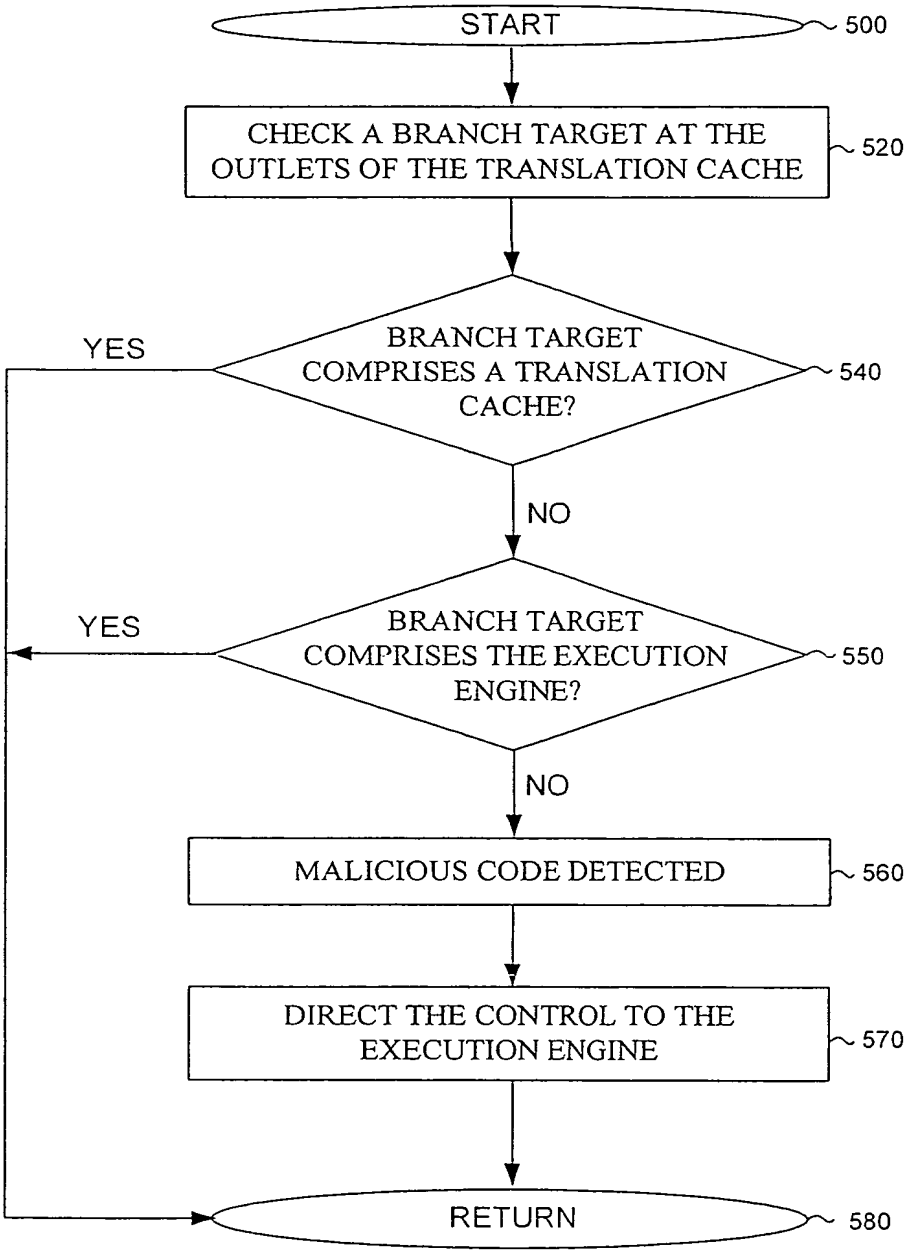
START
500
CHECK A BRANCH TARGET AT THE OUTLETS OF THE TRANSLATION CACHE
520
BRANCH TARGET COMPRISES A TRANSLATION CACHE?
540
YES
NO
BRANCH TARGET COMPRISES THE EXECUTION ENGINE?
550
YES
NO
MALICIOUS CODE DETECTED
560
DIRECT THE CONTROL TO THE EXECUTION ENGINE
570
RETURN
580

FIG. 5

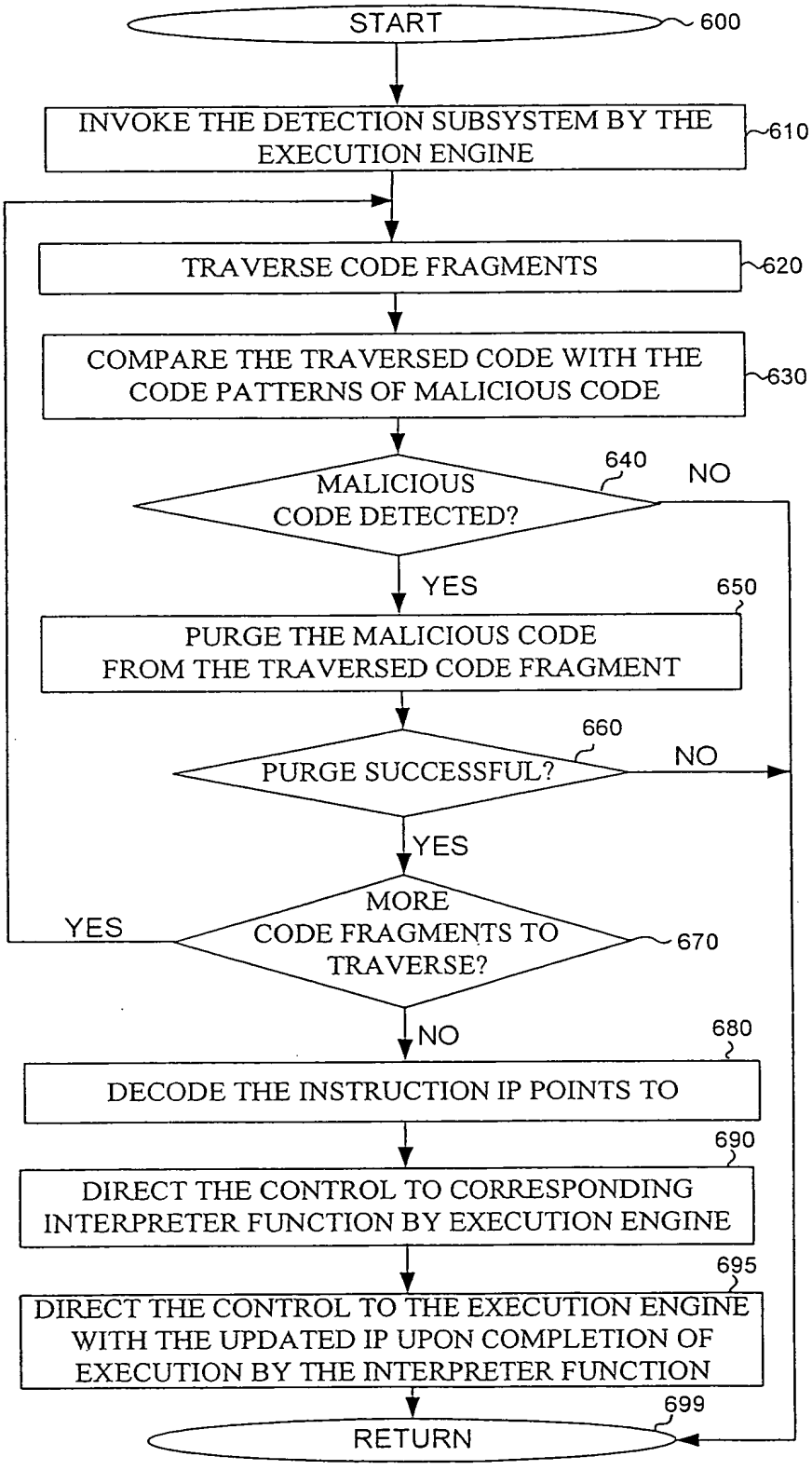
START
600
INVOKE THE DETECTION SUBSYSTEM BY THE EXECUTION ENGINE
610
TRAVERSE CODE FRAGMENTS
620
COMPARE THE TRAVERSED CODE WITH THE CODE PATTERNS OF MALICIOUS CODE
630
MALICIOUS CODE DETECTED?
640
NO
YES
PURGE THE MALICIOUS CODE FROM THE TRAVERSED CODE FRAGMENT
650
PURGE SUCCESSFUL?
660
NO
YES
MORE CODE FRAGMENTS TO TRAVERSE?
670
YES
NO
DECODE THE INSTRUCTION IP POINTS TO
680
DIRECT THE CONTROL TO CORRESPONDING INTERPRETER FUNCTION BY EXECUTION ENGINE
690
DIRECT THE CONTROL TO THE EXECUTION ENGINE WITH THE UPDATED IP UPON COMPLETION OF EXECUTION BY THE INTERPRETER FUNCTION
695
RETURN
699

FIG. 6

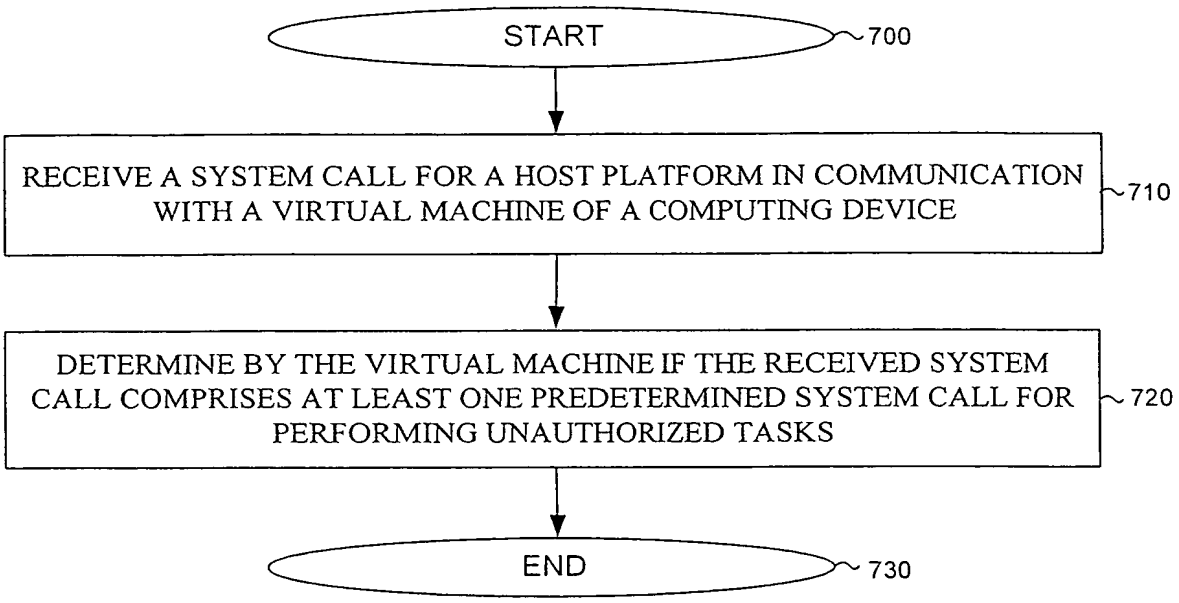
START
700
RECEIVE A SYSTEM CALL FOR A HOST PLATFORM IN COMMUNICATION WITH A VIRTUAL MACHINE OF A COMPUTING DEVICE
710
DETERMINE BY THE VIRTUAL MACHINE IF THE RECEIVED SYSTEM CALL COMPRISES AT LEAST ONE PREDETERMINED SYSTEM CALL FOR PERFORMING UNAUTHORIZED TASKS
720
END
730

FIG. 7A

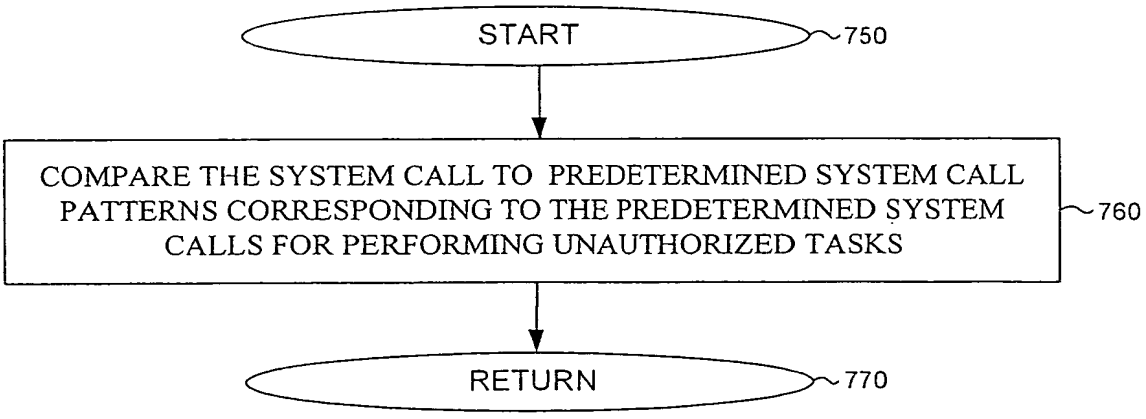
START
750
COMPARE THE SYSTEM CALL TO PREDETERMINED SYSTEM CALL PATTERNS CORRESPONDING TO THE PREDETERMINED SYSTEM CALLS FOR PERFORMING UNAUTHORIZED TASKS
760
RETURN
770

FIG. 7B

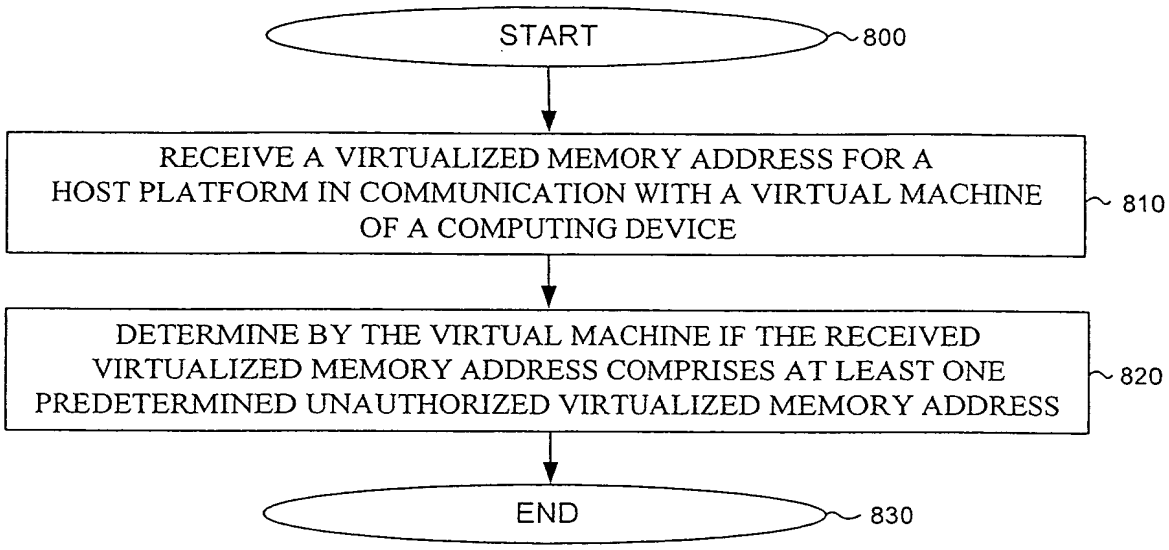
START
800
RECEIVE A VIRTUALIZED MEMORY ADDRESS FOR A HOST PLATFORM IN COMMUNICATION WITH A VIRTUAL MACHINE OF A COMPUTING DEVICE
810
DETERMINE BY THE VIRTUAL MACHINE IF THE RECEIVED VIRTUALIZED MEMORY ADDRESS COMPRISES AT LEAST ONE PREDETERMINED UNAUTHORIZED VIRTUALIZED MEMORY ADDRESS
820
END
830

FIG. 8A

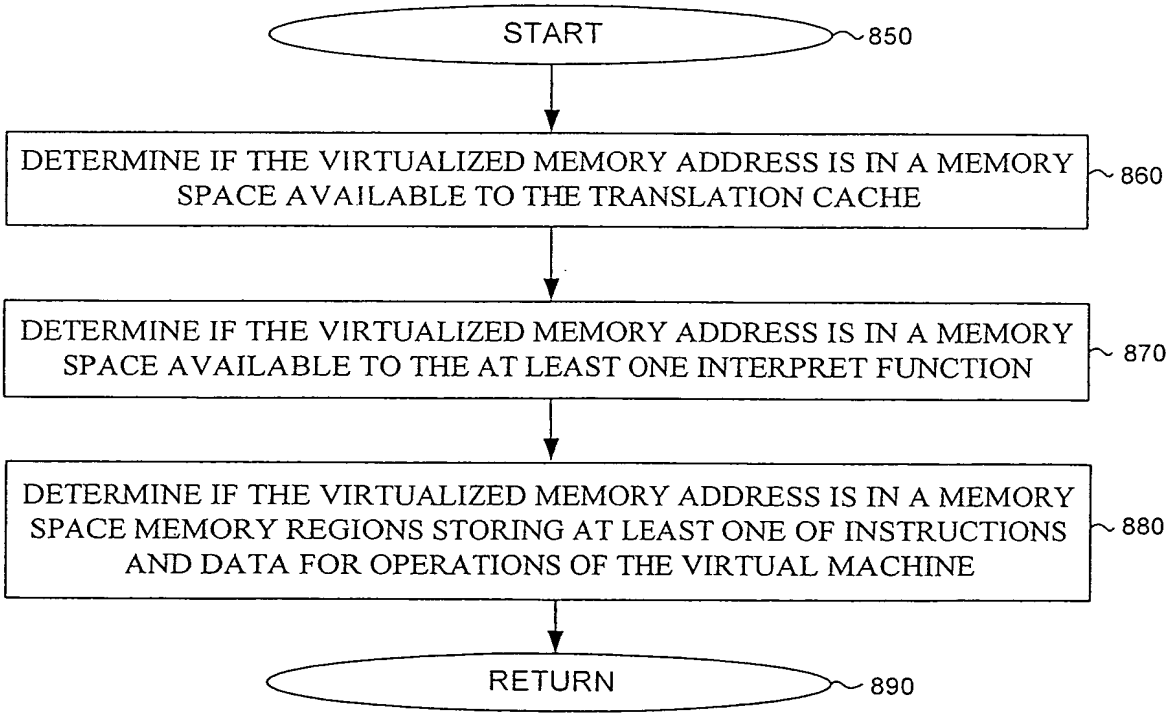
START
850
DETERMINE IF THE VIRTUALIZED MEMORY ADDRESS IS IN A MEMORY SPACE AVAILABLE TO THE TRANSLATION CACHE
860
DETERMINE IF THE VIRTUALIZED MEMORY ADDRESS IS IN A MEMORY SPACE AVAILABLE TO THE AT LEAST ONE INTERPRET FUNCTION
870
DETERMINE IF THE VIRTUALIZED MEMORY ADDRESS IS IN A MEMORY SPACE MEMORY REGIONS STORING AT LEAST ONE OF INSTRUCTIONS AND DATA FOR OPERATIONS OF THE VIRTUAL MACHINE
880
RETURN
890

FIG. 8B